SOFTWARE DEVELOPMENT AND ENGINEERING PRACTICES

Data Access with Entity Framework Core

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

Entity Framework Core is the default way .NET applications talk to a database, and it is deceptively easy to start with: define a class, write a LINQ query, and data appears. The trouble arrives later, in production, as N+1 queries, accidental full-table loads, migration conflicts, and change-tracking surprises. Almost all of that pain traces back to one gap: using EF Core without a clear model of what it is doing on your behalf.

This is a hands-on, practitioner course. It builds that model first: what an ORM actually does, what the DbContext is, and how change tracking works, because every later topic gets easier once you can predict what SQL your code will produce. From there it layers up: modeling entities and relationships, querying with LINQ, saving data safely, evolving the schema with migrations, and finally performance work. Rather than tour every EF Core feature, it goes deep on the workflow and judgment that keep real applications fast and their data correct. Every module ends with a lab and builds on the one before.

Who Should Attend

- C# developers who use EF Core daily but have never been taught it properly
- Back-end developers building APIs or web applications on .NET with relational data
- Developers moving from raw ADO.NET, Dapper, or older Entity Framework to EF Core

Developers new to C# should take *Object-Oriented Programming with C#* first.

Prerequisites

- Working proficiency in C#
- Basic SQL: tables, keys, and simple queries
- Comfortable running a .NET project locally

What You Will Learn

- Explain how EF Core maps objects to tables and what the DbContext and change tracker do
- Model entities, relationships, and constraints with conventions and the fluent API
- Write LINQ queries that translate to efficient SQL, including projections and filtering
- Save data correctly: change tracking, transactions, and concurrency handling
- Evolve a schema safely with migrations, in development and in production
- Diagnose and fix the common performance problems: N+1, over-fetching, and tracking overhead

Course Outline

Day one: the model in your head, and the model in the database

- How EF Core Actually Works
  - What an ORM buys you and what it costs
  - The DbContext: a unit of work, not a global database handle

  - Change tracking: how EF Core knows what to save
  - Lab: set up the course project, inspect the SQL EF Core generates, and predict it before you look
- Modeling Entities and Relationships
  - Conventions, data annotations, and the fluent API, and when to use each
  - One-to-many, many-to-many, and one-to-one relationships done correctly
  - Keys, required fields, indexes, and value conversions
  - Lab: model a realistic multi-entity domain and verify the schema it produces
- Querying with LINQ
  - How LINQ becomes SQL, and where that translation breaks down
  - Filtering, ordering, paging, and projecting with Select
  - Loading related data: eager, explicit, and lazy loading, and their tradeoffs
  - Lab: write a set of increasingly demanding queries and review the SQL each one produces

Day two: writing data, changing schemas, and staying fast

- Saving Data Safely
  - Inserts, updates, and deletes through the change tracker
  - Transactions and when you need to manage them yourself
  - Optimistic concurrency: detecting and resolving conflicting edits
  - Lab: implement a full write workflow, then break it with a simulated concurrency conflict and fix it
- Migrations and Schema Evolution
  - Creating, reviewing, and applying migrations
  - Migrations on a team: merge conflicts and keeping databases in sync
  - Deploying schema changes to production without downtime surprises
  - Lab: evolve your domain model through several migrations, including one that requires a data fix
- Performance and Production Habits
  - Finding the slow parts: logging, query plans, and the N+1 problem
  - AsNoTracking, split queries, compiled queries, and pagination that scales
  - When to drop to raw SQL, and how to do it without losing safety
  - Lab: profile a deliberately slow data layer, find the three worst problems, and fix them

Extended Version

The three-day version keeps the same gradient and adds depth for larger systems:

- Advanced modeling: inheritance mapping, owned types, and shadow properties
- Testing data access: the in-memory trap, SQLite, and integration tests against a real database
- EF Core in architecture: repositories, unit of work, and where the abstractions earn their keep
- A capstone: take a realistic application's data layer from naive to production-ready, with measured before-and-after performance